DBT Learning Notes – Data Build Tool

What is it? - A transformation workflow.

Why use it?

* Benefits with SE best practices like version control, modularity, portability, CI/CD and documentation.
* Dbt helps taking care of the materialization.
* Reduce the time queries take – leveraging **metadata** to find long-running models and use **incremental models**.
* DRYer code by macros, hooks and package management (what are these three???).

Important Features:

* SQL files can contain Jinja, a lightweight templating language. Using Jinja in SQL provides a way to use control structures in your queries. For example, if statements and for loops. It also enables repeated SQL to be shared through macros.
* Macros?
* Ref function?
* Auto-generate documentation
* Auto testing
* Package management – allow you to use and publish repositories so others can re-use them.
* Seed file?
* Snapshot?

Two ways to use dbt:

* dbt Cloud – managed by dbt team for infrastructure, making your life easier, but $ involved.
* dbt Core – open-source tool, manually setup and locally maintain it. Installed through command line.

My own practice:

I tried integrating one dbt initialization with my Snowflake trial account in my exiting project (streaming\_data\_lakehouse\_lab), the process is easy to follow. Some of the critical points include:

1. Other than using pip to install dbt-core, you will also need to install the adapter for the database you are trying to connect to.
   1. Here is the snowflake reference: <https://docs.getdbt.com/docs/core/connect-data-platform/snowflake-setup>
   2. <https://docs.getdbt.com/docs/core/pip-install>
2. When you run the dbt init command for the first time, its interactive cli will ask your snowflake connection information.
   1. When providing account name, you just need the <account-name> part in “<account-name>.snowflakecomputing.com”
3. After the dbt init commands completes, there will be a profile.yml file created under your user directory, in which the database connection info is stored. You can update this file later to change targets.
4. Meanwhile, there will be a dbt\_project.yml file created in your project folder. This file is critical to setup the project.
5. It seems like the table/view created in target DBMS has the same name as the script name.
   1. In the following experiments, I need to understand how we can assert the database and schema of the target. I don’t think we will need to create multiple dbt projects to perform the operations on each database.
6. Packages - Following ChatGPT, I was trying to use a dbt macro to re-create the schema.yml file. Here are some new learnings.
   1. Command I was trying to run: dbt run-operation generate\_model\_yaml --args '{"model\_name": "customers"}'
   2. Faced error that no package – then realized that to run a macro I need to install packages.
   3. Command: dbt deps (To test whether I have the package)
   4. Added info in yml file: A screenshot of a computer

      AI-generated content may be incorrect.
   5. Rerun Command: dbt deps – this will install that dbt-labs/codegen
   6. Later found that the command ChatGPT suggested was out-of-date, so checking: <https://github.com/dbt-labs/dbt-codegen/tree/0.13.1/#generate_model_yaml-source> to find out the right arguments to use.
      1. <https://hub.getdbt.com/dbt-labs/codegen/latest/>
7. <https://docs.getdbt.com/guides/bigquery?step=14> 🡪 Followed this tutorial but using dbt cli to test with creating tables/views in BigQuery.

2025-02-27 video notes:

1. 5 Tips to better work on dbt projects.
   1. Staging layer
      1. Modularity
   2. CTE
      1. Grouping logic
      2. Operate as a pass-through <https://discourse.getdbt.com/t/ctes-are-passthroughs-some-research/155>
   3. Use directories
      1. In dbt\_project.yml file you can refer the schema for the directories you defined.
      2. You can just run dbt models within one directory
      3. Keep your code DRY.
   4. Create a style guide
      1. How you name your columns?
      2. How to collaborative code?
      3. Add them in your readme file.
      4. <https://docs.getdbt.com/best-practices/how-we-style/0-how-we-style-our-dbt-projects>
   5. Don’t optimise for fewer lines
      1. New lines are cheap, brain power is expensive.
      2. Readability
2. Add raw “sources” to dbt project
   1. It’s essentially just .yml files.
   2. Default under model directory, but not exist by default.
   3. <https://docs.getdbt.com/docs/build/sources>
   4. Always add select \* from final at the last line of the model.
   5. Practice Notes:
      1. I tried to setup two different schemas as the target for different folders in models, but it didn’t work well. The problem is that dbt always append default schema and the +schema value to formulate new schemas 🡪 I followed instructions from ChatGPT to introduce a macro to solve the problem.
      2. Macro is easy to work with, you just need to create a .sql file with your macro function under your macro folder and then run dbt clean to ensure the cache is cleared.
      3. I also faced some challenges with the privilege of the running role and the target/source databases, which requires some preparation to grant the right access to the right users.
3. To show documentations generated by dbt, what I need to do is to run the following commands.
   1. dbt docs generate
   2. dbt docs serve – this command raises a server so I can see the documentations.
4. The real query is generated under target/compiled folder, where you can debug.

2025-02-28

1. Custom schema
   1. Option A: in model file, config jinja, overwrite the value of schema.
   2. Option B: in dbt\_project.yml, +schema:<in\_schema\_name>.
      1. But you need to create a macro to overwrite the schema.

2025-03-07

1. Custom schema test:
   1. Very useful feature
   2. User needs to create a macro with prefix “test\_” under macro folder.
   3. User can overwrite existing tests.
   4. User doesn’t need to run `dbt deps` to capture the test model update.
   5. User can check targets file to find the compiled sql command.